ST443 Group Project

Group 11

Candidate Numbers:

1. **47565**
2. **45795**
3. **48272**
4. **56266**
5. **55974**

December 2022

Contents

1 Real Data Classification Problem

1.1 Introduction

The aim of Task 1 is to apply modern statistical machine learning techniques on a real data classification problem based on distinguish the T regulatory cells (TREG) from the conventional CD4 positive T cells (CD4+T).

1.2 Exploratory Data Analysis

1.2.1 Structure of Data

On observing the structure of the data we see that our dataset has 5471 rows and 4124 columns. Clearly, number of observations is greater than the number of features. Since, we are having high-dimensional problem we would need to reduce the number of features for building machine learning models to make it computationally feasible.

1.2.2 Distribution of Labels

Since it is a classification problem so we start off by checking the distribution of labels in our data. We can see that we have 2065 TREG cells and 3406 CD4+T cells. Hence, it is a balanced dataset.

1.2.3 Finding Top 5 Genes in CD4+T and TREG Cells

Through this analysis we are able to determine the variation of genes in TREG and CD4+T cells across mean, variance, and frequency(percentage of cells which have non-zero levels) to determine the significance of genes in our two classes of cells.

	TREG			CD4+T		
Attributes	Mean	Variance	Frequency	Mean	Variance	Frequency
1	RPL41	MT.ND4L	HSP90AA1	RPL41	MT.ND4L	RPL41
2	HSP90AA1	HSPA1A	RPL41	HSP90AA1	HSPA1A	HSP90AA1
3	RPS26	S100A4	RPS26	RPS26	EEF1G	RPL27A
4	PFN1	MT.ATP8	MT.ND4	PFN1	MT.ATP8	RPS21
5	HLA.A	RGS1	FTL	RPS26	RPS4Y1	RPS26

Table 1: Top 5 genes in CD4+T and TREG Cells

1.2.4 Plotting Top 10 Genes as per Mean Difference

We calculate the absolute difference between the mean of Top 10 Genes across two cells and then see the barplot to see the variation in means across the genes. Because there are obvious differences in the mean values of the expression levels of these ten mRNAs in the two types of cells, it is possible to pay attention to these mRNAs when distinguishing the cell.

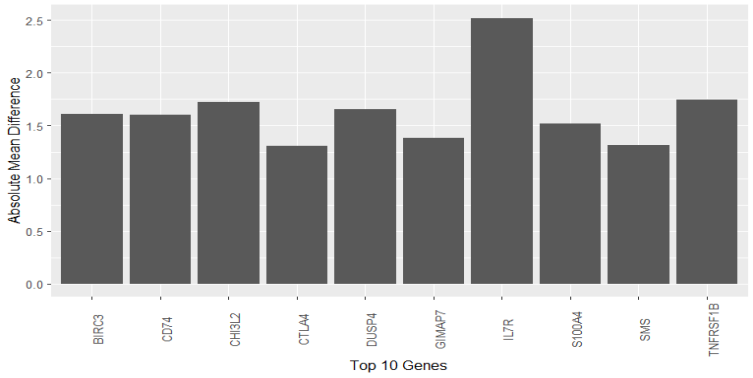

Figure 1: Top 10 Genes as per Mean Difference

1.3 Train-Test Split

We split 75% of our dataset into training set and the rest of it into a testing set. It is important to split our dataset in order to prevent overfitting of model and produce incorrect results.

1.4 Variable Selection Using Boruta

Since, we are dealing with high dimensional dataset so we need to reduce the number of features in order to build machine learning model. If we have too many features included, the model may become too complicated and the computational complexity is going to increase. To improve efficiency and accuracy, variable selection is of great importance. Among different variable selection methods, Boruta is a feature selection algorithm which uses Random Forest as an underlying model. We are able to determine the important features as random forest uses variable importance to determine the significance of variable in the model. By performing feature selection, we are able to reduce the number of dimensions to 217 from 4124 and hence ease up the process of building our model.

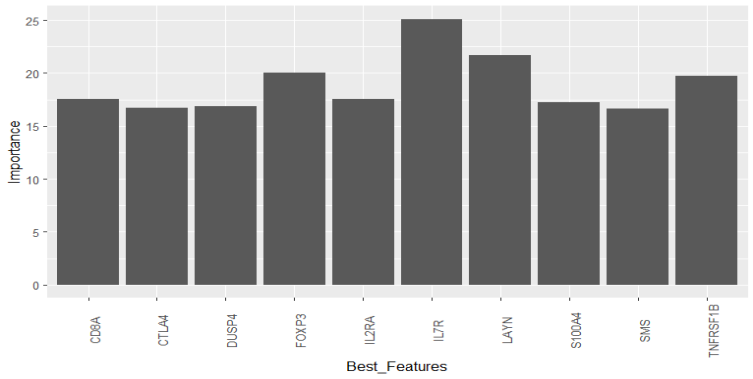

Figure 2: Top 10 features using Boruta

1.5 Classification Approaches

We use 7 different models for solving the classification problem and then we aggregate the result of these classifiers by taking the mean of the binary responses to see whether we can get better result on aggregating various different classifiers.

1.5.1 LDA

Linear Discriminant Analysis plays an important role in extracting classification information and compressing feature space dimension. With the help of packages in R, we can easily apply the LDA function and get the coefficients of variables that have been selected, which give us a matrix describing the linear transformation required to first sphericise the covariance and then project the data onto the hyperplane spanned by the sphericised class centroids. The projected plot clearly indicates that we have successfully classify the data into two separate groups. Next, predict function is used to predict labels based on our constructed test set. In this regard, confusion matrix is established to evaluate the performance of our model. We can conclude that the outcome is not bad due to a high accuracy of 96.07%.

1.5.2 Quadratic Discriminant Analysis

Quadratic Discriminant Analysis has a similar principle with LDA but QDA has a non-linear boundary. Through the same procedure, we classify our data points and get the confusion matrix. This model gives an accuracy rate of 94.34%.

1.5.3 Logistic Regression

The second classification method that we use is logistic regression which uses a sigmoidal function to predict the probability of a datapoint belonging to a particular class. To explain it further, we take advantage of generalized linear model with binomial family to fit our selected training data and predict the outcome by test data. Same as before, confusion matrix can be built, from which we can see an accuracy rate of 95.67%.

1.5.4 K Nearest Neighbour

The K-Nearest Neighbor classifier classifies a new data point by a majority voting of the class label of k nearest training data point to the initial data point. As we do not know the best choice of k, which can be regarded as a tuning parameter, so make use of the

10-fold cross validation method to choose the optimal k. As per the accuracy metric we get the best value of k=3. We the build the confusion matrix to get the accuracy of our method as 90.95%.

1.5.5 Lasso Logistic Regression

Lasso Logistic Regression performs its own variable selection procedure on the entire dataset instead of using the best features obtained via Boruta to predict the response. We know that as lambda increases, the coefficients gets reduced to zero and hence achieve variable selection but the misclassification error rate gets increased, which requires us to have some kind of tradeoff between the two effects. In order to deal with this tradeoff, we get the coefficients corresponding to the mse which is within one standard error of the lowest mse using the best lambda, use the fitted model to predict the test data and calculate the confusion matrix through which we are able to get the accuracy as 95.75%.

1.5.6 Random Forest Classifier

Random Forest classifier is an ensemble learning technique which combines weak learners built out of decision trees to get agrregated and produce robust predictions. The decision trees are built by splitting the dataset based on features which leads to lowest cross entropy value or gini index value until we reach the final leaves. At the leaves, majority voting of class labels takes place to form the final label of our datapoint. We fit our model using the training data and predict on test data to form the confusion matrix through which we get an accuracy of 95.75%.

1.5.7 Support Vector Classifier

Support Vector Classifier is a classification technique mostly used for performing binary classification. It uses the concept of a margin to separate the two class of datapoints. The distance of the nearest datapoints (called support vectors) to the margin is maximised to form the maximal margin classifier and classify the datapoints. Since, there are chances of misclassification of datapoints due to the presence of outliers, so we make use of cross validation to determine the optimal cost associated with misclassification. We get the optimal value of c as 0.001 and we use this value to build the model and predict on the test set to get the confusion matrix which results in an accuracy of 95.91%.

1.5.8 Best Approach- Aggregating the Models

Here, we aggregate the results of various models used for classification to get more robust predictions on the labels of our datapoint. We convert the factor labels produced by various models into binary labels (0 and 1) and take maximum votes to determine the final class of our datapoint. We get an accuracy of 96.7% by using this technique.

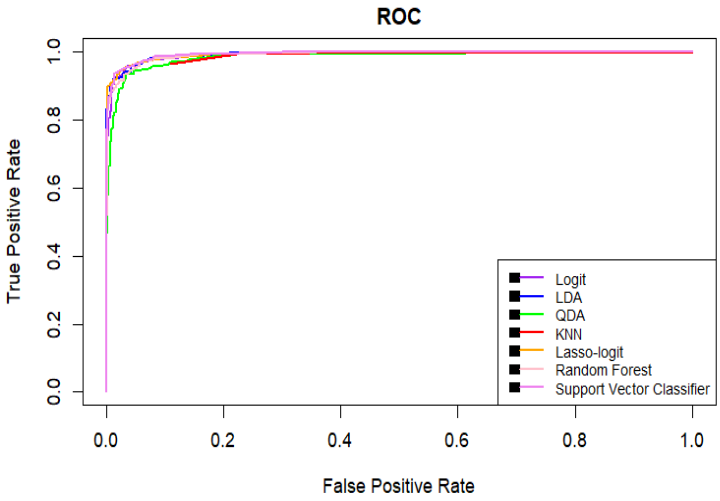

Figure 3: ROC Curve of 7 models

		True Label	
		CD4+T	**TREG**
LDA Predictions	**CD4+T**	762	36
	TREG	14	459
QDA Predictions	**CD4+T**	740	36
	TREG	36	459
Logistic Predictions	**CD4+T**	746	25
	TREG	30	470
KNN Predictions	**CD4+T**	771	110
	TREG	5	385
Lasso Logistic Predictions	**CD4+T**	753	31
	TREG	23	464
RandomForest Predictions	**CD4+T**	758	36
	TREG	18	459
Logistic Predictions	**CD4+T**	750	26
	TREG	26	469
Aggregate Model	**CD4+T**	761	27
	TREG	15	468

Table 2: Confusion Matrix of all models

1.6 Conclusion

In terms of accuracy, we get the best predictions by aggregating the result from all 7 different models. Hence, we can claim that aggregation produces the best result for the classification problem at hand. While in terms of stand-alone models, LDA performs the best as per the accuracy metric.

2 Graphical Model Estimation

2.1 Method Introduction

In this part, we use the graphical model estimation to study the conditional dependence structure among several different variables. We build a network which consists of all variables, which are called nodes, and a number of edges, which we obtain by connecting a subset of the nodes. The basic principle is that two variables are connected if and only if the covariance between two variables conditional on the remaining variables does not equal to zero. For such kind of pairs of nodes, we can construct the so-called edges. Here we talk about two approaches: Node-wise Lasso approach and Graphical Lasso approach. We are going to compare the performance of these approaches in different situation.

2.1.1 Node-wise Lasso Approach

Assuming the variables follow a multivariate normal distribution, for each node, one can regress it on the remaining variables.

$$X_j = \sum_{\substack{1\leq l\leq p \\ l\neq j}} \beta_{j,l} X_l + \varepsilon_{j,l}$$

After choosing a sequence of suitable tuning parameters, we can fit these lasso models to get the coefficients β. Two rules to estimate the edge set are as follows:

I: both $\beta_{j,k}$ and $\beta_{k,j}$ are not equal to zero (Node-wise Lasso 1)
II: either $\beta_{j,k}$ or $\beta_{k,j}$ is not equal to zero (Node-wise Lasso 2)

2.1.2 Graphical Lasso Approach

Assuming the variables follow a multivariate normal distribution with covariance matrix $\Sigma \in \mathbb{R}^{p\times p}$. The rule is that the nodes j and k are connected by an edge if and only if the (j,k)-th entry of the inverse covariance matrix does not equal to zero. After choosing a suitable tuning parameter, we can get the estimated inverse covariance matrix and see whether the (j,k)-th entry equals to zero.

2.2 Selection of Optimal Tuning Parameters

2.2.1 Selecting the Best Tuning Parameters in Node-wise Lasso

We apply 10-fold cross-validation to find the best tuning parameters (*lambda*) for Node-wise Lasso 1 and Node-wise Lasso 2. For each regression $X_j = \sum_{\substack{1\leq l\leq p \\ l\neq j}} \beta_{j,l} X_l + \epsilon_{j,l}$, we try 51 λs in [0,0.5], with step length equal to 0.01. Every time we run one regression with one λ, we can calculate the number of correctly estimated coefficients equal to the number of $\hat{\beta}_{j,l} \in \{(\hat{\beta}_{j,l} \neq 0 \bigcap \Theta_{j,l} \neq 0) \bigcup (\hat{\beta}_{j,l} = 0 \bigcap \Theta_{j,l} = 0)\}$, and the correctly estimated rate equals to the number of correctly estimated coefficients/(p-1). The best λ for regression i is the one that maximizes the average correctly estimated rate under 10-fold cross-validation. The best λ sequence consist of all the best λs for p regressions which are the best tuning parameters for both Node-wise Lasso 1 and 2.

2.2.2 Select the Best Tuning Parameter in Graphical Lasso

For Graphical Lasso, 10-fold cross-validation is also applied to select the best tuning parameter. Still, we try 51 λs in [0,0.5], with step length equals 0.01. We can get one estimated result for every λ and also we calculate the misclassification error rate which equals to :

$$MER = 1 - \frac{\sum_{i,j=1}^{p} \mathbb{1}\{\hat{\Theta}_{i,j} \in \{(\hat{\Theta}_{i,j} \neq 0 \bigcap \Theta_{i,j} \neq 0) \bigcup (\hat{\Theta}_{i,j} = 0 \bigcap \Theta_{i,j} = 0)\}\}}{p(p-1)}$$

We select the λ that can minimize the average misclassification error rate under 10-fold cross validation.

2.3 Simulation Process

Firstly, we make one simulation according to the instruction and the methods of determining the best tuning parameters described above with p=75, n = 500, sparsity = (P(0)=0.9, P(0.5) = 0.1). What we still need to explain here is how we draw the ROC curve, especially for Node-wise lasso 1 and 2. For Node-wise lasso 1 and 2, if we do not regulate λs, the ROC curve will be very strange. To alleviate it, we come up with the idea that we construct a set of λ sequence around the optimal λ sequence, that is, for each λ, we choose 500 smaller values from 0 to λ and 500 larger values from λ to 1 equidistantly. By constructing a λ sequence set with equidistant corresponding elements and calculating the TPR and FPR for each sequence of λ, we finally get smooth ROC curves. For Graphical Lasso, we choose different 1001 λ from [0,1] with step length equaling 0.001, calculate the TPR and FPR for each λ

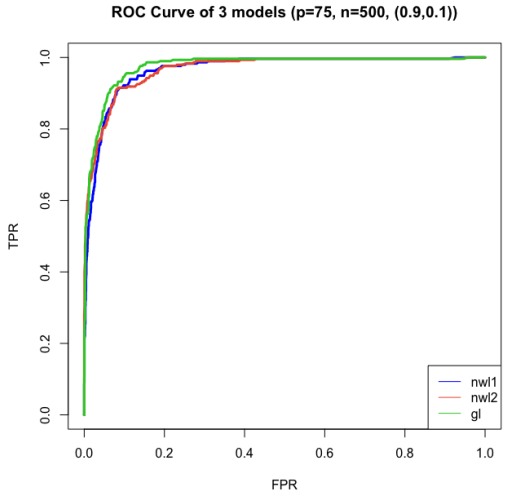

Figure 4: ROC Curve of 3 models

Model	AUROC	FNR	FPR	MER
NWL1	0.959	0.746	0.001	0.080
NWL2	0.983	0.397	0.008	0.049
Glasso	0.987	0.238	0.021	0.047

Table 3: Performance of 3 models

and draw the ROC curve. In addition to the ROC Curve we also compute the AUROC, FNR, FPR, and MER with the best tuning parameters for all 3 models.

It is clear that Graphical lasso has the largest AUROC and the lowest FNR and MER, while in terms of FPR, Node-wise Lasso 1 performs better.

2.4 Different Choices of n, p and Sparsity Structure

Here we plan to see if we change the number of variables, samples and the sparsity structure of Θ, how the outcome will change.

2.4.1 Keep p and sparsity constant, change n

We first set p = 75 and sparsity = (0.9, 0.1) unchanged and set n = 300, 500, and 600.

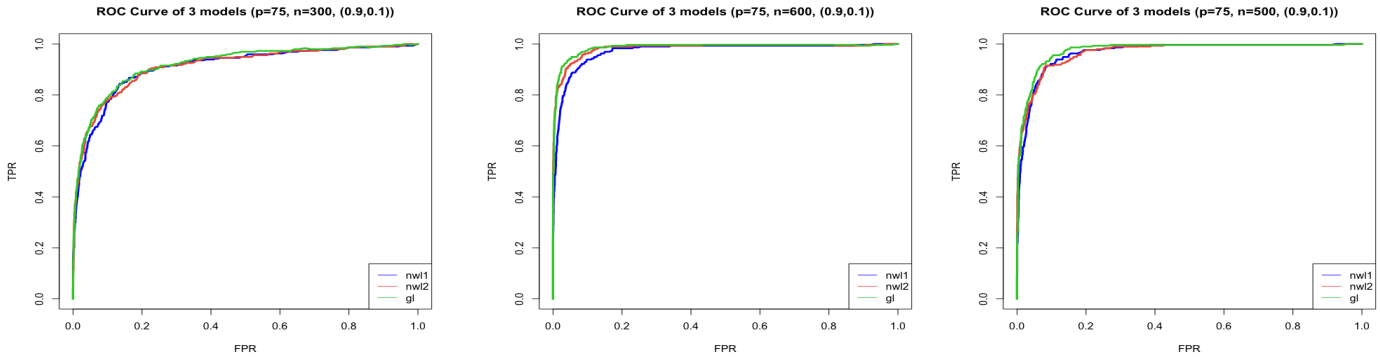

Figure 5: ROC Curve of models with changed p

Model	$n = 300$	$n = 500$	$n = 600$
Node-Wise Lasso 1	0.903	0.959	0.976
Node-Wise Lasso 2	0.902	0.963	0.983
Graphical Lasso	0.916	0.968	0.989

Table 4: AUROC changing n

It is clear that with the increase of n, the AUROC of 3 models all models rise and Graphical Lasso always has the largest AUROC for different n. Therefore, in terms of AUROC, all models have better performance with larger n, and Graphical Lasso has the best performance. Then, we replicate the procedure 100 times for different n, and see how FNR, FPR and MER change.

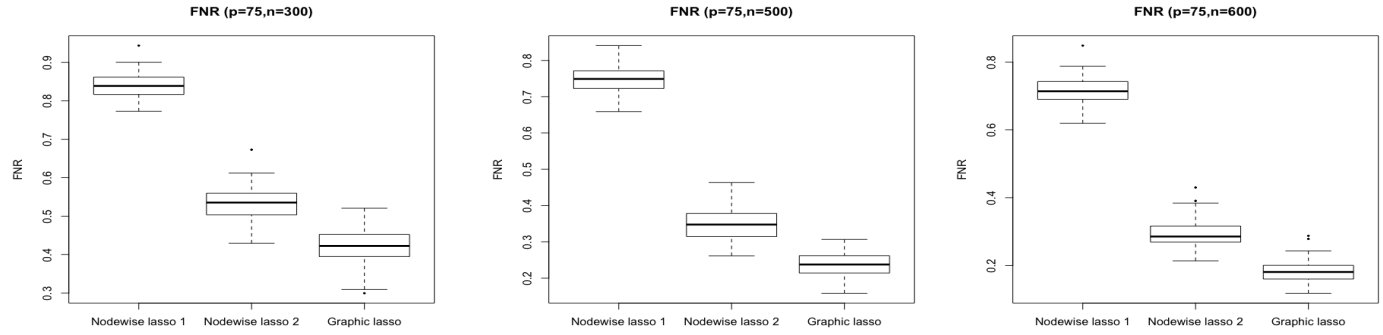

Figure 6: FNR changing n

Figure 6 demonstrates that with the increase of n, the media of FNRs and their variances decline. What's more, Nodewise Lasso 1 performs bad in terms of FNR, which is over 0.7. The other two models perform much better and Graphical Lasso has the lowest FNR.

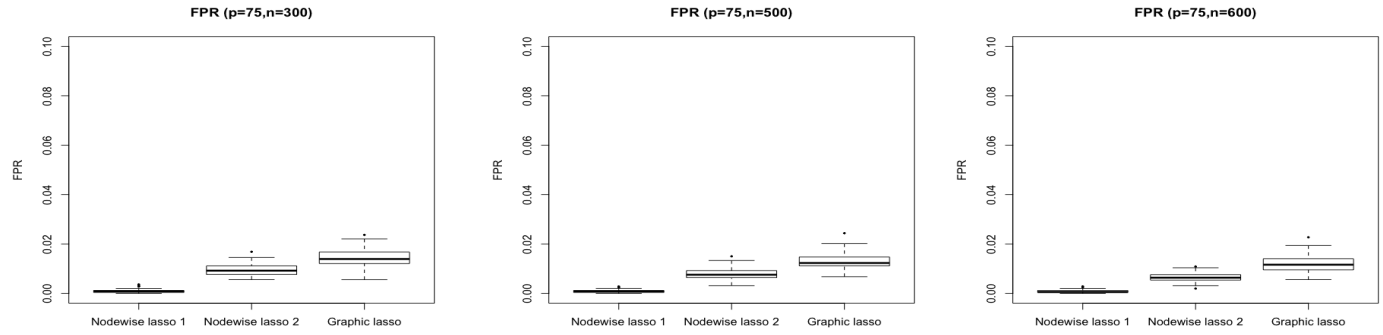

Figure 7: FPR changing n

Figure 7 shows the FPRs for different n. It is clear that 3 models all have better performance than that of FNR: the median of FPRs are around 0.01, and the variance is also very small. Node-wise Lasso 1 gives us the lowest FPR, while Graphical Lasso performs worst. Also, the FPRs drop with the rise of n, but not as significantly as FNRs.

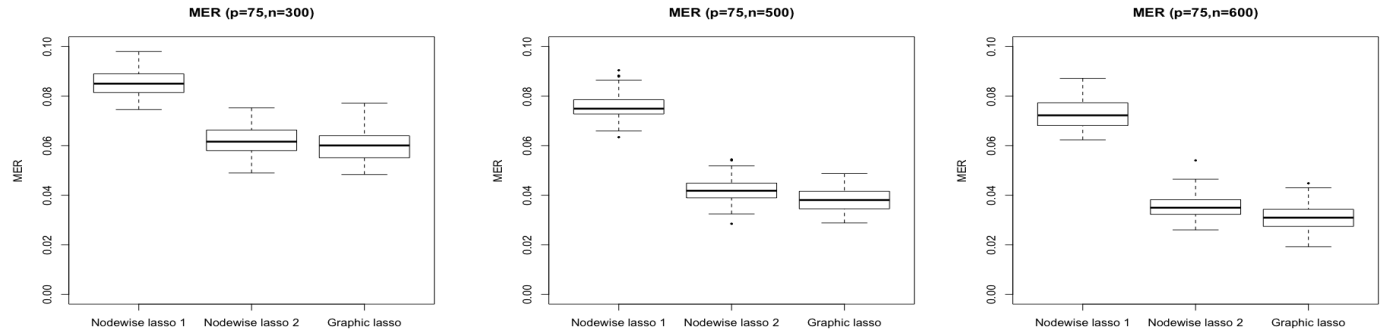

Figure 8: MER changing n

Figure 8 illustrates the change in MERs. With the increase of n, MERs go down, and the gap between Node-wise Lasso 1 and the other two becomes more significant. Node-wise Lasso 1 has the highest MER which is around 0.07-0.08, while Graphical Lasso still performs the best with 0.03-0.04 MER when n = 500 and 600.

	Mean FNR			Mean FPR			Mean MER		
$n =$	300	500	600	300	500	600	300	500	600
Node-wise Lasso 1	0.840	0.747	0.716	0.001	0.001	0.001	0.085	0.076	0.073
Node-wise Lasso 2	0.532	0.349	0.292	0.009	0.008	0.006	0.062	0.042	0.035
Graphical Lasso	0.421	0.236	0.183	0.014	0.013	0.012	0.060	0.038	0.031

Table 5: FNP, FPR and MER changing n

Table 5 summarizes the performance of three models with changing n. When n grows larger, the performance of all 3 models improved, and Graphical Lasso has the best FNR and MER, while Node-wise Lasso 1 performs best in terms of FPR.

2.4.2 Keep n and sparsity constant, change p

We first set n = 500 and sparsity = (0.9, 0.1) unchanged and set p = 50, 75, and 100.

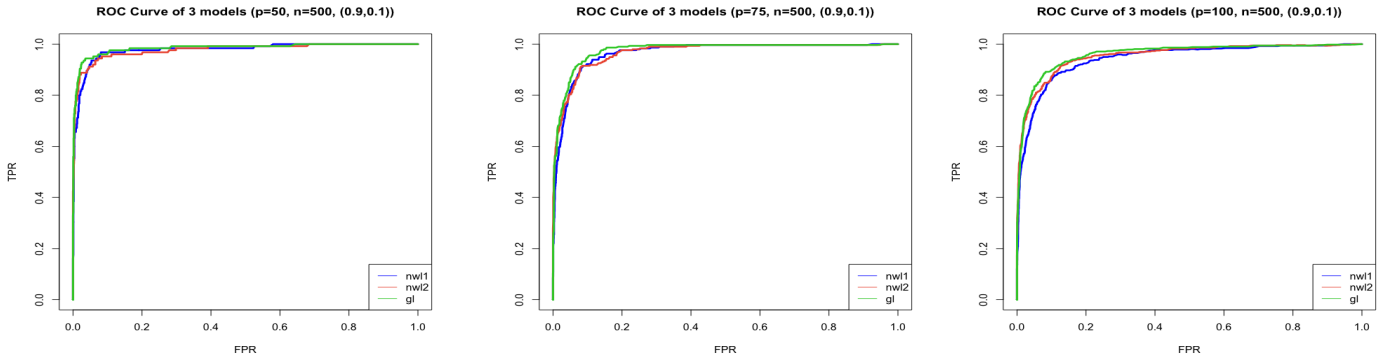

Figure 9: ROC Curve of models changing p

Model	$p = 50$	$p = 75$	$p = 100$
Node-Wise Lasso 1	0.987	0.959	0.950
Node-Wise Lasso 2	0.987	0.963	0.955
Graphical Lasso	0.993	0.968	0.965

Table 6: AUROC changing p

It is clear that with the increase of p, the AUROC of all 3 models become smaller. Now, we replicate the procedure 100 times for different p, and see how FNR, FPR and MER change.

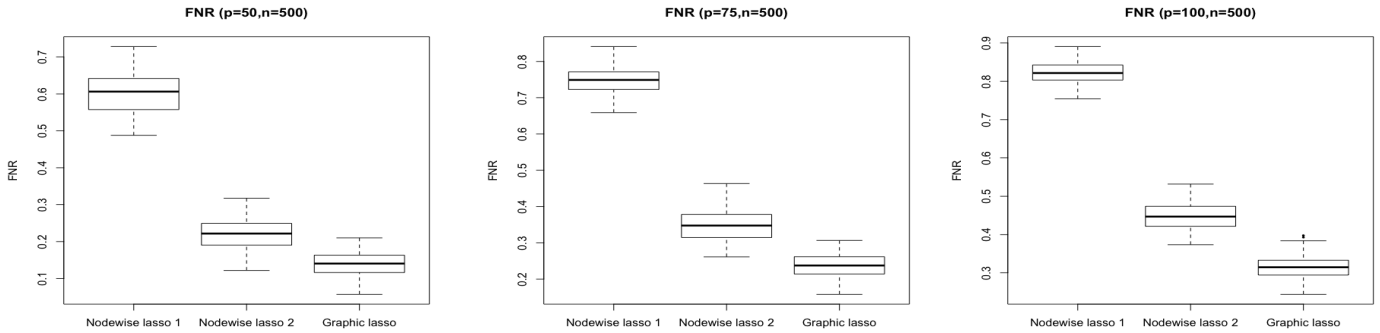

Figure 10: FNR changing p

Figure 10 demonstrates that with the increase of p, the median FNR go up and the variance in FNR go down. Graphical Lasso has the lowest FNR and Node-wise Lasso 1 has the highest FNR.

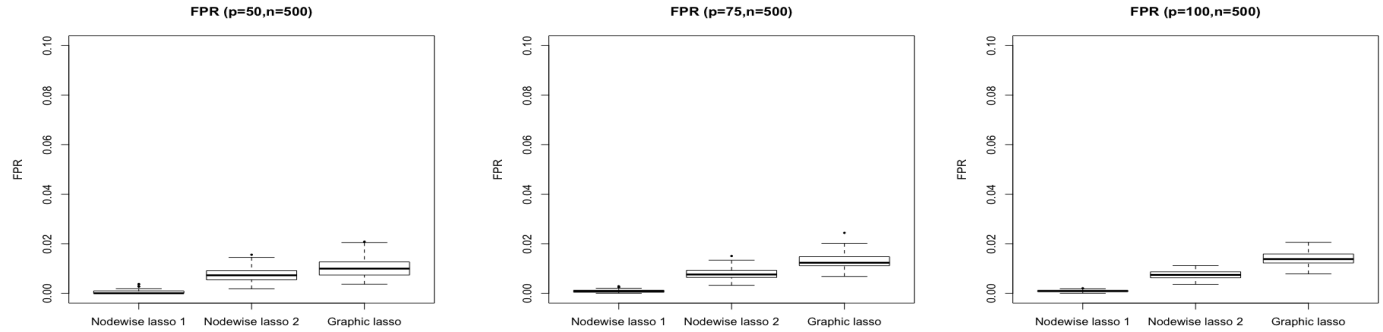

Figure 11: FPR changing p

Figure 11 shows the FPR for different n. The FPR do not change a lot when p rises. The FPR values of the three models are very low, among which FPR of Node-wise Lasso 1 is the lowest while that of Graphical Lasso is the highest. Although in Node-wise Lasso 2 the FPR becomes smaller when change p from 75 to 100, these two values are relatively small and that's probably because of the randomness of the simulation.

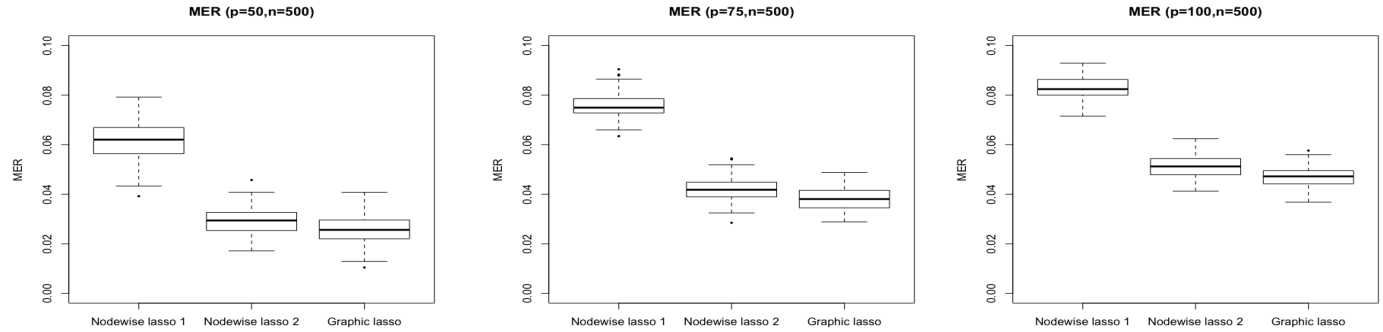

Figure 12: MER changing p

Figure 12 illustrates that with the increase of p, MER increases. Graphical Lasso has the smallest MER, but it is not much different from MER of Node-wise Lasso 2. Node-wise Lasso 1 has the largest MER.

	Mean FNR			Mean FPR			Mean MER		
$p =$	50	75	100	50	75	100	50	75	100
Node-wise Lasso 1	0.601	0.747	0.821	0.001	0.001	0.001	0.061	0.076	0.083
Node-wise Lasso 2	0.219	0.349	0.448	0.008	0.008	0.007	0.029	0.042	0.051
Graphical Lasso	0.140	0.236	0.316	0.010	0.013	0.014	0.026	0.038	0.047

Table 7: FNP, FPR and MER changing p

Table 7 summarizes the performance of three models with changing p. We can see that with the increase of p, the performance of all 3 models become worse, and mean of FPR is insensitive to the change of p. In addition, the relative performance among the models is the same as in the situation of changing n, Graphical Lasso still has the lowest FNR and MER, while Node-wise Lasso 1 has the lowest FPR.

2.4.3 Keep n and p constant, change sparsity

We first set n = 500 and p = 75 unchanged and set sparsity = (0.9,0.1), (0.8,0.2), (0.6,0.4)

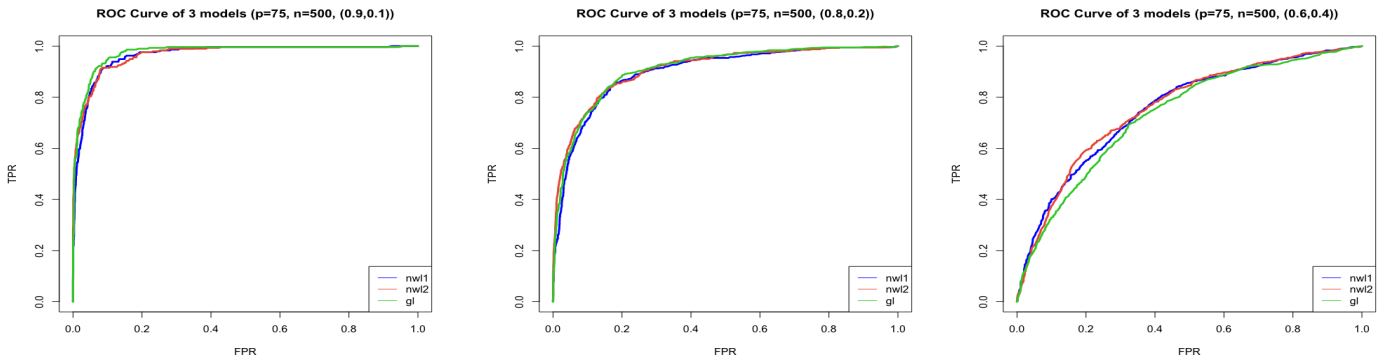

Figure 13: ROC Curve of models changing sparsity

Model	(0.9,0.1)	(0.8,0.2)	(0.6,0.4)
Node-Wise Lasso 1	0.959	0.919	0.772
Node-Wise Lasso 2	0.963	0.919	0.767
Graphical Lasso	0.968	0.925	0.738

Table 8: AUROC changing sparsity

Figure 13 and Table 8 show that when the Θ matrix becomes not that sparse, the AUROC of 3 models decrease, and when the the Θ matrix is sparse, Graphical Lasso performs best. However when it is not sparse, Graphical Lasso becomes the worse models, and Nodel-wise Lasso 1 performs best. Then, we replicate the procedure 100 times for different sparsity, and see how FNR, FPR and MER change.

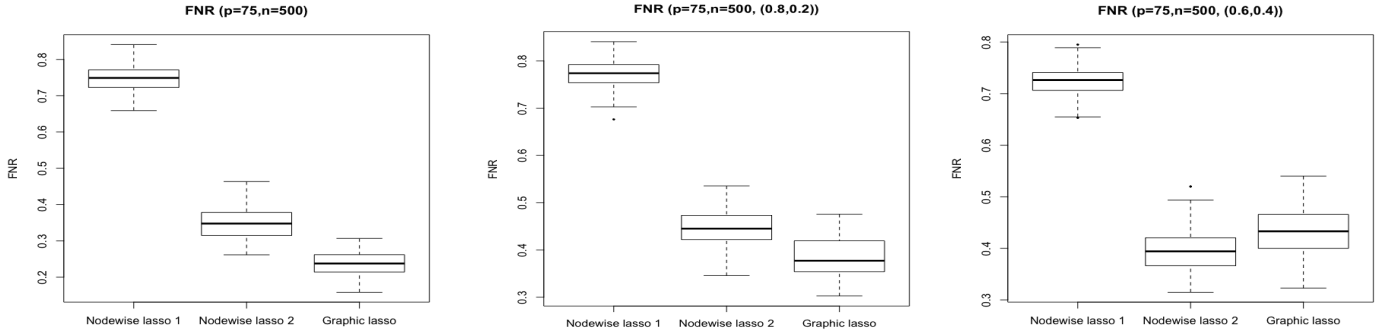

Figure 14: FNR changing sparsity

Figure 14 demonstrates that with the decrease of sparsity, the median FNR of Node-wise Lasso 1 and 2 fluctuate and the median FNR of Graphical Lasso increases. What's more, Nodewise Lasso 1 still performs bad in terms of FNR. When the Θ matrix is sparse, Graphical Lasso performs the best, but when it is not sparse, Node-wise Lasso 2 has the lowest FNR.

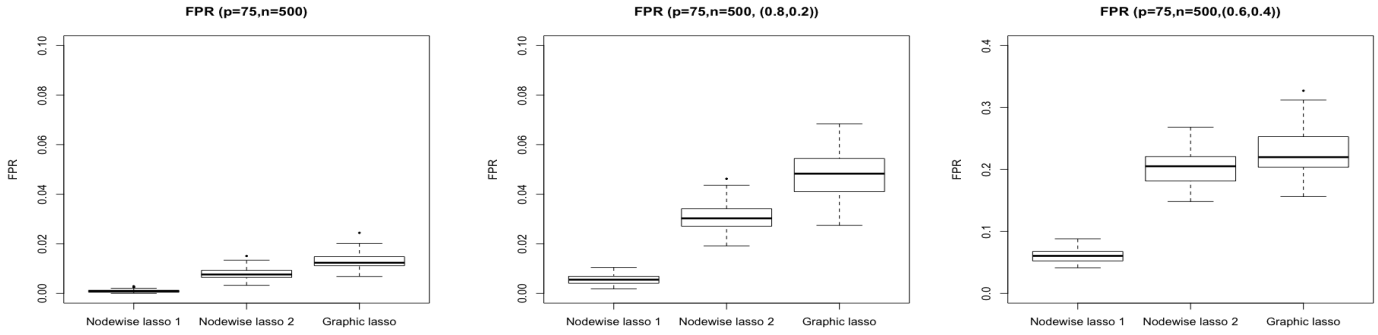

Figure 15: FPR changing sparsity

Figure 15 shows that with the decrease of sparsity, mean and median FPR and their variance rise sharply. Node-wise Lasso 1 still has the lowest FPR even if the sparsity changes.

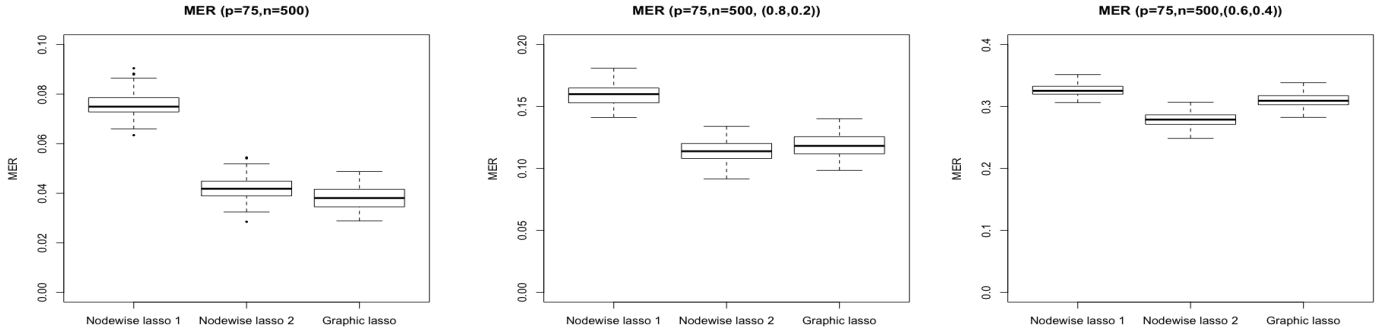

Figure 16: MER changing sparsity

Figure 16 illustrates that with the decrease of sparsity, the median MER climb up, while their variances become smaller. Also, the differences among 3 models become smaller. In additional, when Θ matrix is not sparse anymore, the model with the lowest MER changes from Graphical Lasso to Node-wise Lasso 2.

	Mean FNR			Mean FPR			Mean MER		
$P(\Theta_{i,j}=0)=$	0.9	0.8	0.6	0.9	0.8	0.6	0.9	0.8	0.6
Node-wise Lasso 1	0.747	0.773	0.723	0.001	0.006	0.061	0.076	0.159	0.326
Node-wise Lasso 2	0.349	0.448	0.396	0.008	0.031	0.203	0.042	0.114	0.280
Graphical Lasso	0.236	0.385	0.430	0.013	0.048	0.227	0.038	0.119	0.310

Table 9: FNP, FPR and MER changing sparsity

Table 9 summarizes the performance of three models with changing sparsity. When the sparsity decrease, the mean FPR and MER become worse, while the median of FNR of Node-wise Lasso increase first and then decrease. Moreover, the Graphical Lasso's performance falls dramatically, and Node-wise Lasso 2 outperforms Graphical Lasso in terms of FNR and MER.

2.5 Summary of the findings

When p and sparsity remain unchanged and n becomes larger, the model performs better, but the model is not very sensitive to n. Overall, Graphical Lasso performs the best, and Node-wise Lasso 1 performs the worst.

- In terms of AUROC, the AUROC of the three models becomes larger as n increases, and the values are all greater than 0.90. Among them, Graphical Lasso performs the best, and Node-wise Lasso 1 performs the worst.

- In terms of FNR, as n increases, the mean and variance both show a downward trend. Among them, Graphical Lasso performs best. Node-wise Lasso 1 has the highest FNR value, and as n increases, the FNR decline trend is not obvious, which shows that the model has a large error under true Positive.

- In terms of FPR, as n increases, the mean and median decreases, but the change is not significant. The FPR values of the three models are very low, indicating that the models have a very small error under true Negative, among which Node-wise Lasso 1 performs best, and Graphical Lasso performs the worst.

- In terms of MER, as n increases, the mean and meidan decreases. Among them, Graphical Lasso performs best, but it is not much different from Node-wise Lasso 2. Node-wise Lasso 1 performs poorly, and with the increase of n, the downward trend of MER is not obvious, indicating that the accuracy of model is low.

When n and sparsity remain unchanged and p becomes larger, the model performs worse, but the change is not drastic, indicating that the model is not very sensitive to changes in p. Overall, Graphical Lasso performs the best, and Node-wise Lasso 1 performs the worst.

- In terms of AUROC, the AUROC of the three models becomes smaller as p increases, and the values are all greater than 0.95. Among them, Graphical Lasso performs the best, and Node-wise Lasso 1 performs the worst.

- In terms of FNR, with the increase of p, the mean shows an upward trend, and the variance shows a downward trend. Among them, Graphical Lasso performs best and Node-wise Lasso 1 performs the worst.

- In terms of FPR, as p increases, the mean shows slight variation, but the change is not significant. The FPR values of the three models are very low, indicating that the errors of the three models under true Negative are very small, among which Node-wise Lasso 1 performs best, and Graphical Lasso performs the worst.

- In terms of MER, as p increases, the mean increases. Graphical Lasso performs best, but it is not much different from Node-wise Lasso 2. Here, Node-wise Lasso 1 performed poorly.

When n and p are constant and the matrix Θ is more sparse, the model performs better, but the models are more sensitive to sparsity. When the sparsity is large, Graphical Lasso performs best, and when the sparsity is small, the three models have their own advantages and disadvantages.

- In terms of AUROC, when the matrix is sparse, Graphical Lasso performs best, and Node-wise Lasso 1 and Node-wise Lasso 2 perform similarly. But when the matrix is not sparse, Node-wise Lasso 1 performs best, and Graphical Lasso performs worst.

- In terms of FNR, with the decrease of sparsity, the mean,median and variance of Graphical Lasso show an upward trend. Graphical Lasso performs best when sparsity is high, and Node-wise Lasso 2 performs best when sparsity is low. The FNR values of Node-wise Lasso 1 are all high, which indicates its performance is poor.

- In terms of FPR, as the sparsity decreases, the mean and median of Graphical and Node-wise Lasso 2 increase significantly, indicating that the FPRs of these two models are more sensitive to sparsity, and the FPR value of Node-wise Lasso 1 is the least.

- In terms of MER, as the sparsity decreases, the mean amd median values increase, and the MERs of the three models are more sensitive to sparsity changes. Graphical Lasso performs better when the sparsity is high, but in low sparsity, Node-wise Lasso 2 performs best. Node-wise Lasso 1 performs worst in both cases.